Harnessing the Latent Space: From Steering Vectors to Model Calibrators for Control and Trust

Nishant Subramani CMU
CMU: Carnegie Mellon University, Language Technologies Institute
nishant2@cs.cmu.edu

Abstract

Language models have changed from unreliable text generators to highly-capable large models with trillions of parameters. Capability increases come hand-in-hand with increases in scale, making understanding the internal representations of models more challenging. Since millions of users increasing rely on language models to interact with external tools or make decisions in medium or high-stakes scenarios, we need to establish control over model behavior and know when to trust model outputs. In this paper, we discuss our contributions on harnessing the latent spaces by proposing steering vectors for *control* and developing latent space-based model calibrators for *trust*. Together, our contributions help demystify the latent spaces of language models and offer new insights into how to harness model internals to build more trustworthy language technology.

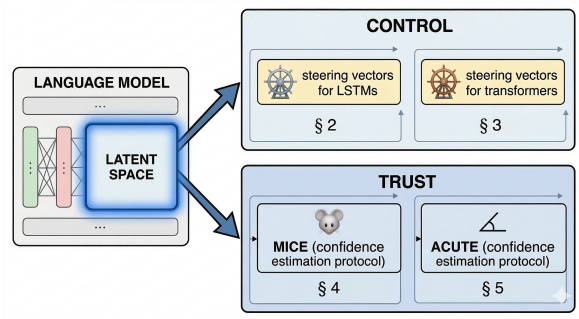

Figure 1: Our contributions on harnessing the latent spaces of language models: §2 and §3 focus on control, proposing steering vectors for the first time for LSTMs and transformer-based models. §4 and §5 focus on trust, building model-internal confidence estimators to assess confidence of language model output generations.

1 Introduction

Neural network language models (LMs) have evolved from small, unreliable text generators to very large models capable of solving complex reasoning tasks (Peters et al., 2018; Radford et al., 2019; Groeneveld et al., 2024; Yang et al., 2025; Team et al., 2025, *inter alia*). Despite the vast capability increases, analyzing the internal representations of trillion-parameter models is challenging. Due to this, the NLP community has increasingly treated models as black boxes, neglecting understanding the inner-workings of models. Even though large language models (LLMs) are scaled to millions of users, increasingly interact with external tools (Qu et al., 2024), and make decisions in medium and high-stakes scenarios (Thirunavukarasu et al., 2023), we rely by-and-large on simple behavioral observation (Hendrycks et al., 2021; Srivastava et al., 2023; Liang et al., 2023, *inter alia*). As a community, we must build fundamental understanding of the inner-workings of models and operationalize the internal representations of LLMs. We need to establish **control** over model behavior to ensure safety and alignment and establish confidence estimation mechanisms which can accurately adjudicate **trust**.

We present four threads of research aimed to demystify and harness the latent spaces of language models. To achieve model control, we show that LSTM-based language models can be minimally steered for exact generation (§2). We then adapt to transformer-based models in §3, showing both fine-grained and coarse-grained control via exact and concept-based steering. Shifting to trustworthiness, we build model-internal confidence estimators (MICE) to calibrate LLM generations in tool-calling scenarios (§4). Lastly, we broaden the framework to new model families and tasks by proposing activation-based confidence, utility, and trust estimators (ACUTE; §5). Together, our contributions offer actionable recipes to harness model internals to build more controllable, well-calibrated, and trustworthy language technologies.

Proceedings of The Big Picture v2: Crafting a Research Narrative, pages 119–130

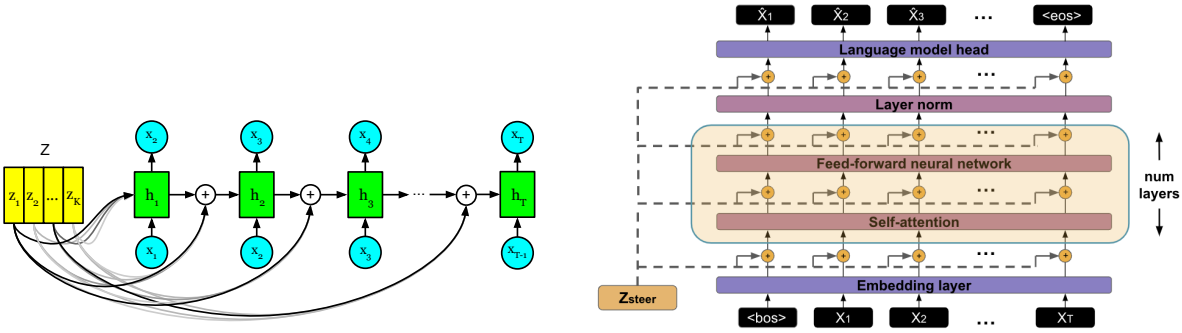

Figure 2: Here we show how the steering vector z_{steer} can be injected into an LSTM-based language model (left) and a transformer-based one (right). On the left, Z is shown to have a larger dimension than the model dimension. If dim(Z) equals the model dimension, $K = 1$, and thus there is just one vector z_1.

2 Control: Steering Vectors for LSTMs (Subramani et al., 2019)

We focus on control, specifically trying to answer one key question:

Key Question 1

Can LSTM-based language models be steered to generate a desired sequence exactly without updating a single parameter?

2.1 Prior Work

In 2018, the transformer architecture proposed in Vaswani et al. (2017) had yet to fully permeate the language model landscape and long short-term memory models (LSTMs; Hochreiter and Schmidhuber (1997)) were still the predominant architecture for language modeling. These language models were unreliable text generators. However, they have the potential to learn useful representations, so LMs started to be seen as general-purpose encoders (Dai and Le, 2015; Peters et al., 2018; Devlin et al., 2019, *inter alia*). Precise control of language model output, on the other hand, remained far out of reach, primarily due to the low quality of the underlying language models of the time.

2.2 Our Contributions

We explore whether language models could be used as *general-purpose decoders*, something that we now take for granted, but at the time was an unknown. For a pretrained language model to be used a general-purpose decoder, we need (1) to find a continuous-valued sentence representation (a steering vector) that can be fed into the frozen language model, (2) an encoder, likely task-specific, that can convert task inputs into steering vectors, and (3) for those steering vectors to *causally* generate the desired output. At the time, no work had shown this was possible, but now we take this for granted with advances in prompting and decoder-only LLMs. In our work, we explore the possibilities of this in LSTM-based models, before prompting became popular. Specifically, we ask whether LSTM-based models can be steered to generate a desired sequence exactly while keeping the underlying language model frozen.

Background To ask this, we first define the *sentence space* of a recurrent language model. Since the recurrent transition function $f_\theta = \mathbb{R}^d \times V \to \mathbb{R}^d$ defines a dynamical system based on the observations of tokens in a sequence. As a result, the language model embeds a sequence of length T as a $T + 1$ step trajectory in a d-dimensional space, where d is the dimension of the hidden state of the recurrent LM. Next, we parametrize the sentence space into a flat-vector space $\mathcal{Z} \in \mathbb{R}^{d'}$ to better understand the sentence space of the LM.

To map the trajectory of hidden states to a flat vector in $\mathcal{Z}$, we add a bias term $z_{steer} \in \mathcal{Z}$ to the previous hidden and cell state at each time step in the model and optimize z_{steer} to maximize the log-probability of a given sequence. Since we're adding z_{steer} to every hidden and cell state as well as at every timestep, information contained in z_{steer} will not degrade as quickly as if we just intervened at one location at one timestep. Using this formulation, we can go back and forth, from vectors to sequences and vice-versa, and thus design experiments to test whether a frozen model can be steered to generate any sequence of interest.

To map from sequences to steering vectors (forward estimation), we modify the recurrent transi-

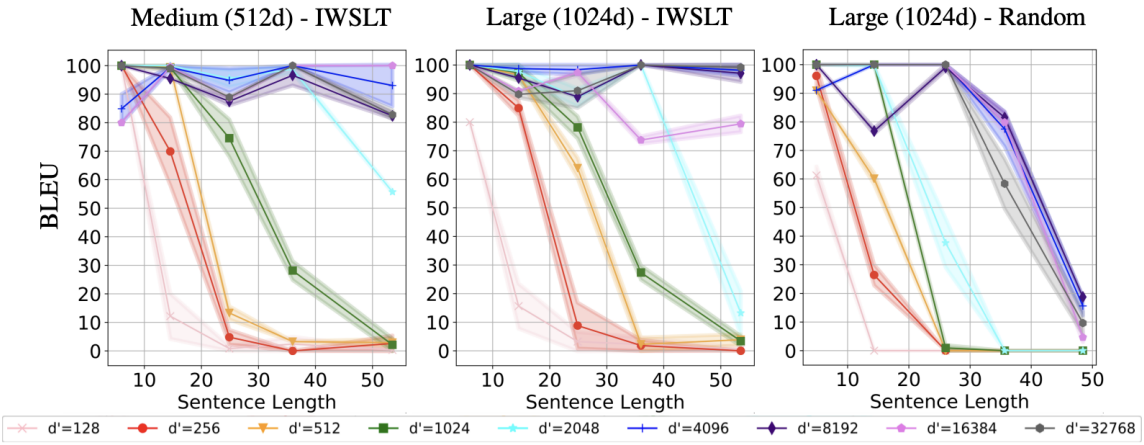

Figure 3: Recovery on IWSLT16 for `medium` (left) and `large` (center) and on random data for `large` (right).

tion function, see Figure 2 for details:

$$h_t = f_\theta(h_{t-1} + z_{steer}, x_t) \quad (1)$$

Here, we assume the dimension of z_{steer} is equal to the model dimension d^*. If this is not the case, we can up or down project z_{steer} without the addition of any parameters. See Subramani et al. (2019) for details. We then optimize z_{steer} to maximize the log-probability of the sequence as mentioned before via any off-the-shelf gradient-based optimization algorithm (*e.g.*, gradient descent, nonlinear conjugate descent, etc.).

To map steering vectors back to sequences (backward estimation), we intervene on a language model by injecting z_{steer} and using beam search to decode starting with a <bos> token. We stop when an <eos> token or 100 total tokens is reached. We measure how well the original sequence matches the generated sequence via three string overlap metrics: token-level exact match, BLEU score (Papineni et al., 2002), and longest prefix match.

Experimental Setup First, we train our own language models on 50M sentences from the English Gigaword corpus (Graff and Cieri, 2003), with a 879k sentence development set and a 878k sentence test set stratified by article publishing date. We use byte-pair encoding with 20,000 merges for a vocabulary of 20,234 subword tokens (Gage, 1994; Sennrich et al., 2016). Our model is a 2-layer language model with LSTM units of three sizes, `small` ($d = 256$), `medium` ($d = 512$), and `large` ($d = 1024$) with shared input and output embeddings (Press and Wolf, 2017), and dropout (Srivastava et al., 2014).

To train the model, we use stochastic gradient descent with Adam with a learning rate of 1e-4 and a batch size of 100 (Kingma and Ba, 2015). To learn steering vectors, we sample 100 randomly selected sentences from the development set as well as 50 sentences from the IWSLT16 En-De translation dataset to measure out-of-distribution generalization (Cettolo et al., 2016). We use nonlinear conjugate gradient (Wright and Nocedal, 1999) for optimization due to the highly non-convex nature of the objective function and use beam search with a width of 5 for backward estimation (Graves, 2012). See Subramani et al. (2019) for more details.

2.3 Takeaways

We can find steering vectors for every sequence that achieve near perfect recoverability (token-level exact match ≥ 0.99) on `large`, offering an avenue for direct causal control. Additionally, we find that larger, better trained models have higher recoverability and longer sequences are harder to recover. One key question is whether our forward estimation procedure operates like a naive compressor without any structure. In other words, does the forward estimation procedure have enough capacity to just encode the entire sequence in the vector without leveraging the language model's internal representations of language? To test this, we create a *random* dataset where we sample from the vocabulary with replacement at random where every token has equal probability. We learn steering vectors for these sequences as well as the out-of-domain IWSLT16 data and measure recoverability. In Figure 3, we show that sequences that are lower entropy under the language model (IWSLT data) are much easier to recover at similar sequence lengths as compared to sequences from the *random* data.

However, even for the very high entropy sequences (ones from *random*), z_{steer} has the capacity to encode sequences up to a token length of 28 nearly perfectly.

Limitations: Finding steering vectors was challenging. Optimization via conjugate gradient methods was very slow and could not be easily GPU-accelerated at the time, reducing adoption. Given how good recovery was for random sequences, a natural follow-up would be to understand what steering vectors encode and whether they could be useful beyond single sequence interventions. There is no guarantee that the steering vector space learned here offers any additional utility. This is precisely what we expand upon and explore in the next section.

2.4 Bigger Picture

At the time, being able to intervene on a language model with a single vector and causally force the model to generate *any* sequence of interest without updating a single parameter was highly surprising. This meant that language models had tremendous potential as universal decoders and steering could open up avenues to move away from task-specific finetuning and replace with inference-time steering. Our work could serve as justification to attempt natural language prompting on better trained, stronger models, which occurred in the years that followed. BERT had just recently come out (Devlin et al., 2019), and while this paper was under review, we learned that BERT rediscovered the classical NLP pipeline (Tenney et al., 2019), hinting that internal structure likely exists in transformer-based language models.

3 Control: Steering Vectors for Transformers (Subramani et al., 2022)

We expand upon control, generalizing steering vectors to transformer-based language models for both *exact steering* and *concept-based steering*. We answer the following questions:

Key Question 2

Can transformer-based language models be steered to generate a desired sequence exactly without any parameter updates?

Injection location	Timestep	BLEU-4
Embedding	all timesteps	33.99
Layer 6 (self attn)	**all timesteps**	**100.0**
Layer 6 (self attn)	**first timestep**	**99.80**
Layer 7 (feed fwd)	**all timesteps**	**100.0**
Layer 7 (feed fwd)	**first timestep**	**99.25**
All layers (self attn + feed fwd)	**all timesteps**	**100.0**
All layers (self attn + feed fwd)	first timestep	91.72
LM head	all timesteps	6.72

Table 1: Sentence recovery for steering vectors when injected into different layers of the transformer model and at multiple timesteps.

Key Question 3

Can extracted steering vectors act as useful representations with which we perform concept-based steering at inference-time?

3.1 Prior Work

Transformer language models started becoming popular, outperforming and largely replacing recurrent models (Devlin et al., 2019; Radford et al., 2019; Raffel et al., 2020). In §2, we showed that LSTM-based LMs could be precisely controlled for short sequences with steering vectors, opening up the potential for them to be used as universal or general-purpose decoders. Here, we explore whether higher-quality transformer-based language models could be more easily and efficiently steered, and thus make better universal decoder candidates. This work began prior to the release of GPT3 (Brown et al., 2020), hence the focus on small transformer-based models rather than LLMs.

3.2 Our Contributions

We coin the term *steering vector*. A vector $z_{steer} \in \mathbb{R}^d$ is a *steering vector* for a sequence x under a model M only if M exactly generates x via greedy decoding when z_{steer} is injected into M.[1]

Background We define a flat-vector space $\mathcal{Z} \in R^{d'}$ for a transformer language model, similar to the recurrent language model from §2. To map the trajectory of hidden states for a sequence

[1]Note that steering vectors need not correspond to an exact sequence. They are commonly now used to steer towards a desired concept or attribute.

$x_1, \ldots, x_T$, we add a bias term $z_{steer} \in \mathcal{Z}$ to the first-timestep at a single layer in the transformer stack after the feed-forward layer, see Figure 2 for details.[2] We also optimize z_{steer} to maximize the log-probability of a given sequence, giving us the ability to map sequences to steering vectors (and vice-versa) and measure recoverability, exactly like in LSTM-based models. This process is more efficient in transformers as compared to LSTMs because z_{steer} is only added at one layer and one timestep. We can up or down project z_{steer} if we want to control the capacity of the steering vector.

Experimental Setup We take the GPT2-117M model and learn steering vectors by sampling sequences from four different genres (movies, books, news, and wikipedia) and stratify them based on length for a total of 256 sequences. Recoverability is measured via BLEU score. We vary where (injection location) and when (injection timestep) to intervene with z_{steer}. For optimization during forward estimation, we use Adam with a learning rate of 1.0 and use greedy decoding to recover sequences. We measure the extent to which steering vectors can be used as representations and compare them with mean-pooled hidden states.

Lastly, we explore whether concept-based steering is possible. We first extract steering vectors for sequences for positive and negative sentiment respectively via the Yelp sentiment dataset (Shen et al., 2017). We propose difference-of-means (DiffMean) steering, which works as follows. First, we use vector arithmetic to take the mean of the positive sentiment steering vectors z_{pos} and the negative sentiment ones z_{neg}. This is then operationalized at inference-time. For example, to steer towards positive sentiment, you add a steering vector $z_{steer} = \lambda(z_{pos} - z_{neg})$ at the timestep and location that those steering vectors were extracted from.[3]

3.3 Takeaways

Our experiments show that fine-grained control via the exact steering of transformer-based language models is much easier and more efficient than the exact steering of LSTMs. Table 1 shows that nearly all sequences are perfectly recovered, even when adding z_{steer} at just the first timestep. As long as z_{steer} is injected in the transformer stack (after the embedding and before the final layer), recoverability remains nearly perfect. Linearly interpolating between steering vectors gives us a glimpse into what the steering vector space looks like. Decoding from these intermediate points reveals structure: the space seems relatively smooth with large clusters corresponding to each of the sequences being interpolated between and a smooth transition in both syntax and semantics when moving from one sequence to another. Cosine distances between steering vectors at middle layers reflect semantic similarity better than mean-pooled hidden states when measured on the semantic textual similarity benchmark (Cer et al., 2017), indicating that steering vectors may be better representations than the ones learned by the underlying language models.

Steering vectors provide coarse-grained control, too. Our experiments on unsupervised sentiment transfer via DiffMean steering on the Yelp sentiment dataset show that a single direction in latent space learned via these steering vectors can flip sentiment reliably. We show two examples in Table 2. For the first time, we show that concept steering at inference-time is possible.

Positive Input	the taste is excellent!
$+0.5 * (z_{neg} - z_{pos})$	the taste is excellent!
$+1.0 * (z_{neg} - z_{pos})$	the taste is excellent!
$+1.5 * (z_{neg} - z_{pos})$	the taste is bitter and bitter taste is bitter taste is bitter
$+2.0 * (z_{neg} - z_{pos})$	the taste is unpleasant.
Negative Input	**the desserts were very bland.**
$+0.5 * (z_{pos} - z_{neg})$	the desserts were very bland.
$+1.0 * (z_{pos} - z_{neg})$	the desserts were very bland.
$+1.5 * (z_{pos} - z_{neg})$	the desserts were very tasty.
$+2.0 * (z_{pos} - z_{neg})$	the desserts were very tasty.

Table 2: Concept steering for sentiment for a positive input sentence (top) and negative input sentence (bottom).

3.4 Bigger Picture

As language model quality started improving, control became an achievable goal. Two months after starting this project, GPT3 came out showing that large pretrained language models had the ability to, at inference-time, be few-shot prompted to solve different tasks. Our work could serve as further justification for more ambitious inference-time based control such as in-context learning, alignment, and persona-based steering. Over the past 5 years, language models became more performant with higher quality representations and concept-based steering took off, operating on the activation

[2] z_{steer} could be added anywhere in the transformer stack and repeated across layers or timesteps, but we found that adding it once at a single layer and first timestep was sufficient.

[3] $\lambda \in \mathbb{R}$ is a constant known as the steering strength.

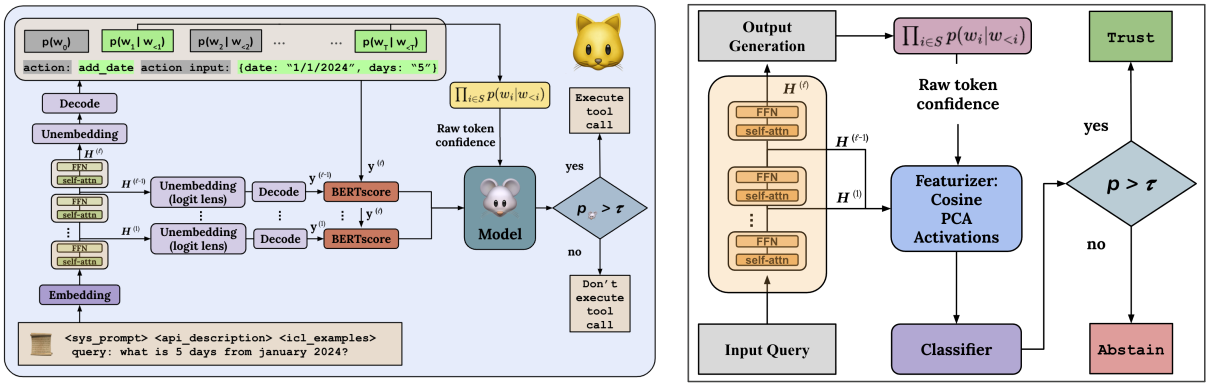

Figure 4: Here we show both the MICE (left) and ACUTE (right) systems to calibrate language model generations. The only major differences between the systems are the featurizers and classifiers used.

space rather than a reparametrized vector space like our steering vector space for inference-time control (Turner et al., 2023; Li et al., 2023; Dunefsky and Cohan, 2025; Arad et al., 2025; Bigelow et al., 2026; Morgulis and Hewitt, 2026; Wurgaft et al., 2026, *inter alia*).

4 Trust: MICE (Subramani et al., 2025a)

We tackle trust by leveraging model internals to try to answer a key question:

Key Question 4

Can we harness the latent spaces of language models to build better confidence estimators for tool-calling agents?

4.1 Prior Work

A confidence estimator is a model that estimates the probability that a different model's output is correct. Since language models have an internal confidence for its output already (*i.e.*, the joint probability of the generated sequence), auxiliary confidence estimators are rarely used. However, raw confidences of language models are known to be poorly calibrated (Desai and Durrett, 2020; Jiang et al., 2021; Zhong et al., 2023). To be well-calibrated, a confidence estimator must be correct approximately as often as it thinks it is (Dawid, 1982).[4]

Confidence estimation in NLP has been studied in tasks such as machine translation (Niculescu-Mizil and Caruana, 2005), semantic parsing (Stengel-Eskin and Van Durme, 2023), and long-form text generation (Band et al., 2024). Calibrating binary classifiers with a single input feature, the raw confidence, is common practice in machine learning with Platt scaling (Platt, 1999), isotonic regression (Barlow, 1972), beta calibration (Kull et al., 2017), and histogram regression estimators with adaptive binning (Nobel, 1996).

Our angle, harnessing the latent spaces of models to build confidence estimation mechanisms, is unique. In fact, there are very few studies that even combine mechanistic interpretability with confidence estimation or trust. Beigi et al. (2024) improves trustworthiness by using contrastive learning on activations and Liu et al. (2025) predicts correctness in question-answering tasks using probes learned on activations.

4.2 Our Contributions

Motivation: We explore whether we can leverage model internals to build a class of model internal confidence estimators (MICE) to better calibrate tool-calling agents. Taking inspiration from the early-exiting and intermediate layer decoding literature (*i.e.*, the observation that a language model can often be decoded from early layers; Geva et al. (2022); Schuster et al. (2022); Belrose et al. (2023); Elhoushi et al. (2024); Yom Din et al. (2024); Merullo et al. (2024)), we theorize that a prediction that is slowly refined through the layers ought to be more trustworthy than one that suddenly appears in the final layer.

MICE: Figure 4 shows MICE on the left. For a given input query q, we pass it through the language model and at each layer i, we use *logit lens* to decode from that intermediate layer (nostalgebraist, 2020), resulting in a candidate generation $y^{(i)}$. Then, we compare how close that intermediate generation is to the final output generation via

[4] Calibration is commonly measured using expected calibration error (ECE; Murphy and Epstein (1967); Naeini et al. (2015)) and Brier Score (Glenn, 1950).

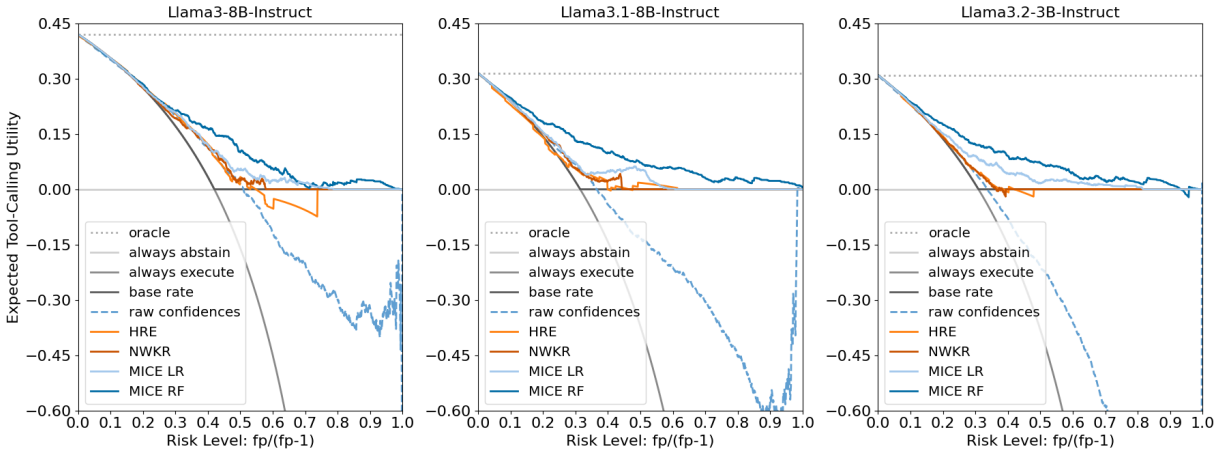

Figure 5: MICE systems outperform baselines on ETCU on the STE test set, especially at high-risk levels.

BERTscore (BERTscore($y^{(i)}, y^{(final)}$); Zhang et al. (2020)) using DeBERTa-xlarge-mnli (He et al., 2021), using those as features to train a correctness classifier. The probability that the classifier assigns to `<correct>` *is* the new calibrated confidence. In practice, we use both BERTscore features and the raw confidence (*i.e.*, the joint probability that the language model assigns to the output sequence) as features to train the classifier.[5]

Measuring Trust: We care about evaluating how *good* a confidence estimator is. This is commonly measured using ECE. However, ECE suffers from two major drawbacks:

1. Cannot distinguish between an oracle and a base rate estimator (*i.e.*, one that just predicts the base rate regardless of correctness)
2. Invariant to the risk level of the task

These drawbacks hinder decision-making utility and thus calibration is necessary but not sufficient for our purpose. We develop our own metric, expected tool-calling utility (ETCU) to solve these drawbacks, offering a better metric with which to evaluate the quality of a confidence estimator.[6] Additionally, we measure calibration error using a recently improved ECE variant called smooth ECE (smECE; Błasiok and Nakkiran (2024)).

Experimental Details: Our experiments use the simulated trial-and-error (STE) dataset, a synthetically generated tool-calling dataset consisting of English-language queries, which call 50 distinct APIs (Wang et al., 2024). We split the data into demonstration (used to construct few-shot examples), training, validation, and test sets consisting of 4250, 1500, 750, and 750 examples each. We 8-shot prompt three LLMs, Llama-3-8B-Instruct, Llama3.1-8B-Instruct, and Llama3.2-3B-Instruct and decode using greedy decoding to generate candidates (Grattafiori et al., 2024). Our MICE classifiers are trained using the training set, hyperparameters are tuned using the validation set, and performance is evaluated on the test set.[7]

4.3 Takeaways

Our experiments confirm that language models are poorly calibrated. Traditional post-hoc recalibration techniques such as histogram regression (HRE) and Nadaraya-Watson kernel regression (NWKR) tend to be highly conservative (Nadaraya, 1964; Watson, 1964), collapsing to base rate estimators with low calibration error, but low decision-making utility. MICE estimators, on the other hand, perform better, maintaining low calibration error, but higher decision-making utility due to a larger spread of probability estimates across examples. In Figure 5, we observe that for tasks with medium or high-risk, both HRE and NWKR remain conservative, always abstaining for all inputs. MICE performs better, correctly increasing its abstention rate as risk-levels increase, while trusting tool-calls some of the time. To quantify how well a confidence estimator does across risk-level settings, we develop an area-under-the-curve (AUC) metric called AUC-ETCU, following Marcum (1960). [8]

To test out-of-domain generalization for tool-

[5] We experiment with two classifiers: a random forest and a logistic regressor.

[6] See Subramani et al. (2025a) for details.

[7] A candidate generation is deemed to be correct if and only if it exactly matches the ground-truth answer.

[8] AUC style metrics are used in many areas of science (Wagner and Ayres, 1977; Geifman et al., 2019; Subramani et al., 2025b, *inter alia*).

	MMLU					APIGen					SCITLDR				
	(↓)	AUC-EURO (↑)				(↓)	AUC-EURO (↑)				(↓)	AUC-EURO (↑)			
estimator	**smECE**	low	med	high	all	**smECE**	low	med	high	all	**smECE**	low	med	high	all
Raw Conf	0.17	0.90	0.71	0.54	0.72	0.22	0.28	0.53	0.80	0.53	0.15	0.36	0.71	**0.92**	0.66
HRE	0.11	0.87	0.72	0.71	0.77	**0.02**	0.82	0.70	0.82	0.78	**0.08**	0.67	0.71	**0.92**	0.77
NWKR	**0.07**	0.90	0.73	0.74	0.79	**0.02**	0.82	0.69	0.82	0.78	**0.08**	0.67	0.71	**0.92**	0.77
ACUTE early act	**0.07**	**0.91**	0.73	0.74	0.79	0.05	0.90	0.82	0.87	0.86	**0.08**	0.69	**0.72**	**0.92**	**0.78**
ACUTE mid act	**0.07**	**0.91**	0.76	0.78	0.82	0.06	0.90	**0.84**	**0.88**	0.87	**0.08**	**0.70**	**0.72**	**0.92**	**0.78**
ACUTE late act	**0.07**	**0.91**	**0.77**	**0.80**	**0.83**	0.07	0.90	**0.84**	**0.88**	0.87	**0.08**	0.69	**0.72**	**0.92**	0.77
ACUTE cosine	0.09	0.90	0.75	0.78	0.81	0.03	0.87	0.77	0.84	0.83	**0.08**	0.68	0.71	**0.92**	0.77
ACUTE pca10	0.08	**0.91**	0.76	0.78	0.82	0.04	0.90	0.82	0.87	0.87	0.09	**0.70**	**0.72**	**0.92**	**0.78**
ACUTE pca20	0.08	**0.91**	0.76	0.77	0.81	0.06	**0.91**	**0.84**	**0.88**	**0.88**	0.09	**0.70**	**0.72**	**0.92**	**0.78**

Table 3: Results on the MMLU test set averaged across all 57 subtasks (left), on the APIGen test subset (middle), and on the SCITLDR dev set (right). All results for all tasks are averaged across the 6 LLMs we test. Lower smECE is better, while higher AUC-EURO is better. **Bold**, underline indicate the best and second best result respectively.

calling, we create a scenario in which new APIs are tested on. We hold out each of the 50 distinct APIs sequentially, resembling 50-fold cross validation and combine predictions across the entire test set. Despite being zero-shot, MICE performs comparably to post-hoc calibration baselines across both smECE and ETCU, while having a larger spread of probability estimates across samples.

Limitations: MICE relies on an auxiliary model to calculate BERTscore features, which can be very expensive. Additionally, we experiment with a single model family, Llama, on a single task, tool-calling. Lastly, ETCU makes one strong simplifying assumption, that a confidence estimator should get a reward of 0 for abstaining regardless of whether the candidate generation was correct or not. We address all of these limitations in §5.

5 Trust: ACUTE (Subramani et al., 2026)

We improve trustworthiness by addressing some limitations of MICE by asking:

Key Question 5

Can we harness the latent spaces of language models to efficiently build better confidence estimators for model generations across a variety of tasks including multiple-choice question answering, tool-calling, and scientific document summarization?

5.1 Our Contributions

ACUTE: We introduce an activation-based confidence, utility, and trust estimation protocol (ACUTE) to appropriately assess the confidence of a language model output. In Figure 4, we show how ACUTE works. First, we take the activations and pass them through a featurizer. Those features are fed into a classifier to predict whether the output generation is correct or not, exactly like MICE. Finally, the probability that the classifier assigns to correct is the new recalibrated confidence. ACUTE removes the reliance on the auxiliary BERTscore model for input featurization.

EURO: We improve upon the ETCU metric from §4 by removing the assumption that abstaining should always get 0 reward by introducing a new general and easily interpretable metric called expected utility renormalized by the oracle (EURO) that balances decision-making utility with calibration. EURO does not make simplifying assumptions, uses a single degree-of-freedom (the normalized net utility of correctly abstaining u_{ca}), and is bounded between an oracle estimator (EURO=1) and anti-oracle estimator (EURO=0). Calculating EURO at different u_{ca} or risk values traces a curve. Measuring the area under that curve provides a score with which to compare confidence estimators called AUC-EURO.[9]

Experimental Setup We apply ACUTE to six new LLMs on three new tasks: multiple-choice question answering (MMLU; Hendrycks et al. (2021)), tool-calling (APIGen; Liu et al. (2024)), and scientific document summarization (SCITLDR; Cachola et al. (2020)).[10] Performance is measured via smECE and AUC-EURO.

[9] See Subramani et al. (2026) for further details, including a detailed derivation in the Appendix of that paper.

[10] Results are averaged across LLMs.

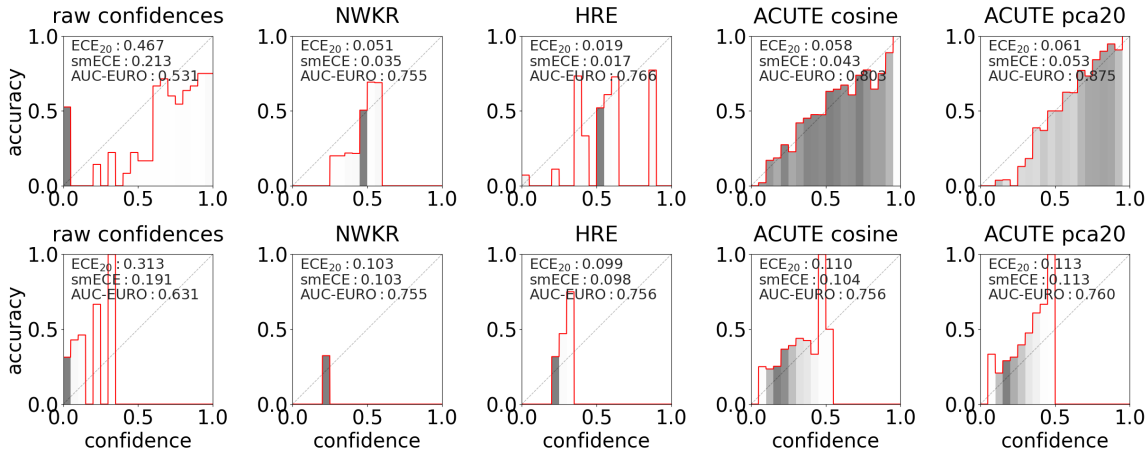

Figure 6: Reliability diagrams for the gemma-3-12b-it model for APIGen (top row) and SCITLDR (bottom row) across 3 baseline estimators and two ACUTE confidence estimators. Darker shading corresponds to higher density of examples in that confidence bin.

5.2 Takeaways

Table 3 and Figure 6 reveals that raw confidences remain poorly calibrated and post-hoc recalibration baselines collapse to base rate estimators as in §4. ACUTE performs well across all three tasks, outperforming baselines on AUC-EURO, while maintaining low smECE. Using activations from later layers (ACUTE late act) and PCA-reduction of all layer activations to 20 components (ACUTE pca20) performed best across all tasks.

5.3 Bigger Picture

Without any post-hoc calibration, language models provide terrible confidence estimates. Most post-hoc calibration methods collapse the often large spread of probability estimates into a much narrower range, reducing calibration error without increasing decision-making utility. Despite these drawbacks, probability estimates with or without post-hoc calibration remain the *de-facto* standard. Our work challenges this by proposing decision-making utility centric confidence estimators that can better adjudicate trust. Further study on this front can help us develop even better confidence estimators and increase the reliability and safety of LLMs. Overall, our work is a small step towards harnessing the latent spaces to appropriately assign trust to LLM outputs.

6 Conclusion

We present four research contributions that demonstrate how we can operationalize the latent spaces of language models for better control and trust. Our work introduces steering vectors for exact and concept-based control on both LSTM- and transformer-based models. Steering vectors provide nearly perfect fine-grained control at inference-time, suggesting that language models could be universal decoders. We propose two methods which harness the latent spaces of models to learn confidence estimators for language model generations to improve trust. Our methods recalibrate model outputs across architectures and tasks effectively, suggesting that the latent spaces contain information to build more reliable and trustworthy language technology, especially in high-stakes scenarios. We hope that our work encourages others to open up the black box and study the latent spaces of language models.

Acknowledgments

We thank the numerous collaborators and authors on each of the individual papers discussed here and both Mona Diab and Nivedita Suresh for feedback on an early version of this work.

References

Dana Arad, Aaron Mueller, and Yonatan Belinkov. 2025. SAEs are good for steering – if you select the right features. In *EMNLP*.

Neil Band, Xuechen Li, Tengyu Ma, and Tatsunori Hashimoto. 2024. Linguistic calibration of long-form generations. In *ICML*.

Richard E Barlow. 1972. Statistical inference under order restrictions: The theory and application of isotonic regression. *(No Title)*.

Mohammad Beigi, Ying Shen, Runing Yang, Zihao Lin, Qifan Wang, Ankith Mohan, Jianfeng He, Ming Jin, Chang-Tien Lu, and Lifu Huang. 2024. InternalInspector i^2: Robust confidence estimation in LLMs through internal states. In *EMNLP Findings*.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor V. Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *arXiv*.

Eric Bigelow, Daniel Wurgaft, YingQiao Wang, Noah Goodman, Tomer Ullman, Hidenori Tanaka, and Ekdeep Singh Lubana. 2026. Belief dynamics reveal the dual nature of in-context learning and activation steering. *Preprint*, arXiv:2511.00617.

Jarosław Błasiok and Preetum Nakkiran. 2024. Smooth ECE: Principled reliability diagrams via kernel smoothing. In *ICLR*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *NeurIPS*.

Isabel Cachola, Kyle Lo, Arman Cohan, and Daniel Weld. 2020. TLDR: Extreme summarization of scientific documents. In *EMNLP Findings*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval*.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Rolando Cattoni, and Marcello Federico. 2016. The IWSLT 2016 evaluation campaign. In *Proceedings of the 13th International Conference on Spoken Language Translation*. International Workshop on Spoken Language Translation.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. *NeurIPS*.

A Philip Dawid. 1982. The well-calibrated bayesian. *Journal of the American statistical Association*, (379).

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained transformers. In *EMNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Jacob Dunefsky and Arman Cohan. 2025. One-shot optimized steering vectors mediate safety-relevant behaviors in llms. In *COLM*.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. 2024. LayerSkip: Enabling early exit inference and self-speculative decoding. In *ACL*.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal archive*.

Yonatan Geifman, Guy Uziel, and Ran El-Yaniv. 2019. Bias-reduced uncertainty estimation for deep neural classifiers. In *ICLR*.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *EMNLP*.

W Brier Glenn. 1950. Verification of forecasts expressed in terms of probability. *Monthly weather review*, (1).

David Graff and Christopher Cieri. 2003. English gigaword corpus. *Linguistic Data Consortium*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 22 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv*.

Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, and 22 others. 2024. OLMo: Accelerating the science of language models. In *ACL*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *ICLR*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *ICLR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, (8).

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *TACL*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Meelis Kull, Telmo Silva Filho, and Peter Flach. 2017. Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research. PMLR.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Inference-time intervention: Eliciting truthful answers from a language model. *NeurIPS*.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher R'e, Diana Acosta-Navas, Drew A. Hudson, and 22 others. 2023. Holistic evaluation of language models. *Annals of the New York Academy of Sciences*.

Jiarui Liu, Jivitesh Jain, Mona Diab, and Nishant Subramani. 2025. Llm microscope: What model internals reveal about answer correctness and context utilization. *arXiv*.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, Rithesh RN, and 1 others. 2024. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *NeurIPS*.

J.I. Marcum. 1960. A statistical theory of target detection by pulsed radar. *IRE Transactions on Information Theory*.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. 2024. Language models implement simple Word2Vec-style vector arithmetic. In *NAACL*.

George Morgulis and John Hewitt. 2026. Subliminal steering: Stronger encoding of hidden signals. *arXiv*.

Allan H Murphy and Edward S Epstein. 1967. Verification of probabilistic predictions: A brief review. *Journal of Applied Meteorology and Climatology*, (5).

Elizbar A Nadaraya. 1964. On estimating regression. *Theory of Probability & Its Applications*, (1).

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. 2015. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, 1.

Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In *ICML*.

Andrew Nobel. 1996. Histogram regression estimation using data-dependent partitions. *The Annals of Statistics*, (3).

nostalgebraist. 2020. Interpreting GPT: The logit lens. Blogpost.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

John Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, (3).

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Jirong Wen. 2024. Tool learning with large language models: a survey. *Frontiers of Computer Science*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. *NeurIPS*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and T. Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *NIPS*.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, and 22 others. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *TMLR*. Featured Certification.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, (56).

Elias Stengel-Eskin and Benjamin Van Durme. 2023. Calibrated interpretation: Confidence estimation in semantic parsing. *TACL*.

Nishant Subramani, Samuel Bowman, and Kyunghyun Cho. 2019. Can unconditional language models recover arbitrary sentences? *NeurIPS*.

Nishant Subramani, Jason Eisner, Justin Svegliato, Benjamin Van Durme, Yu Su, and Sam Thomson. 2025a. MICE for CATs: Model-internal confidence estimation for calibrating agents with tools. In *NAACL*.

Nishant Subramani, Alfredo Gomez, and Mona T. Diab. 2025b. SimBA: Simplifying benchmark analysis using performance matrices alone. In *EMNLP Findings*.

Nishant Subramani, Palash Goyal, Yiwen Song, Mani Malek, Yuan Xue, Tomas Pfister, and Hamid Palangi. 2026. The acute protocol: Operationalizing language model activations for better calibration, utility, and trust. In *ICML*.

Nishant Subramani, Nivedita Suresh, and Matthew Peters. 2022. Extracting latent steering vectors from pretrained language models. In *ACL Findings*.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *ACL*.

Arun James Thirunavukarasu, Darren S. J. Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. 2023. Large language models in medicine. *Nature Medicine*.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. 2023. Steering language models with activation engineering. *arXiv*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

John G. Wagner and James W. Ayres. 1977. Bioavailability assessment: Methods to estimate total area (auc 0-∞) and total amount excreted (ae∞) and importance of blood and urine sampling scheme with application to digoxin. *Journal of Pharmacokinetics and Biopharmaceutics*.

Boshi Wang, Hao Fang, Jason Eisner, Benjamin Van Durme, and Yu Su. 2024. LLMs in the imaginarium: Tool learning through simulated trial and error. In *ACL*.

Geoffrey S Watson. 1964. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*.

Stephen Wright and Jorge Nocedal. 1999. Numerical optimization. *Springer Science*, (67-68).

Daniel Wurgaft, Can Rager, Matthew Kowal, Vasudev Shyam, Sheridan Feucht, Usha Bhalla, Tal Haklay, Eric Bigelow, Raphael Sarfati, Thomas McGrath, Owen Lewis, Jack Merullo, Noah Goodman, Thomas Fel, Atticus Geiger, and Ekdeep Singh Lubana. 2026. Manifold steering reveals the shared geometry of neural network representation and behavior. *Preprint*, arXiv:2605.05115.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv*.

Alexander Yom Din, Taelin Karidi, Leshem Choshen, and Mor Geva. 2024. Jump to conclusions: Short-cutting transformers with linear transformations. In *COLING*. ELRA and ICCL.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *ICLR*.

Ruiqi Zhong, Charlie Snell, Dan Klein, and Jason Eisner. 2023. Non-programmers can label programs indirectly via active examples: A case study with text-to-SQL. In *EMNLP*.